CS530
Authorization

Bill Cheng

http://merlot.usc.edu/cs530-s10

Authorization: Two Meanings

- Determining permission
  - Is principal P permitted to perform action A on object U?
- Adding permission
  - P is now permitted to perform action A on object U
- In this course, we use the first sense

Access Control

- Who is permitted to perform which actions on what objects?
- Access Control Matrix (ACM)
  - columns indexed by principal
  - rows indexed by objects
  - elements are arrays of permissions indexed by action
- In practice, ACMs are abstract (not realizable) objects
  - ACM is huge and sparse
  - ACM is often distributed
  - instantiations
    - ACLs
    - capabilities

Instantiations of ACMs

- Access Control Lists (ACLs)
  - *for each object*, list principals and actions permitted on that object
  - corresponds to rows of ACM
    - can be compacted (null entries removed)
  - e.g., Kerberos admin system
- Capabilities
  - *for each principal*, list objects and actions permitted for that principal
  - corresponds to columns of ACM
  - e.g., Kerberos restricted proxies
    - e.g., I’m authorized to transfer money from A to B
  - it is easy to delegate capabilities
- The Unix file system is an example of...?

Problems

- Permissions may need to be determined dynamically
  - time
  - system load
  - relationship with other objects
    - e.g., can only write to this file if this other file is present
  - security status of host
    - e.g., only administrators are allowed to login if the system is under attack

Problems (Cont...)

- Distributed nature of systems may aggravate this
  - problem with centralized approach is that you have to contact the server to determine permissions on every access, distributed is more efficient
  - ACLs need to be replicated or centralized
    - e.g., yellow pages on Solaris
  - capabilities don’t, but they’re harder to revoke
    - a live object carries capabilities in memory
    - must have a revokation list to be checked when capabilities are presented
- Approaches
  - GAA (next lecture)
  - agent-based authorization
    - mobile piece of code that acts on behalf of a principle

Agent-Based Authorization

- When object created on a host H, agent Q created along with it
  - agent aids in making authorization decisions
- Agents distributed to clients
  - either directly, or through agent server
- Client on host G instantiates agent for principal P, submits it to H as Q/P@G
  - Q acts on behalf of P at G
- Advantages:
  - dynamic evaluation of policies
  - distributed control
  - ease of administration
  - granularity specific to an object

Agent-Based Authorization (Cont...)

- Relieves scaling issues with ACLs
- Q is typically mobile code and data
  - needs to be integrity-protected
  - may be confidentiality-protected
  - agent environment on H must be trusted

Revocation in Agent-Based Systems

- Timeout-based
- Harder for malicious agents
  - hosts must send CRLs (certificate revocation lists) to other hosts and/or principals
  - must maintain their own CRL to restrict or deny incoming agents